Surrogate Modeling of Electrical Machine Torque Using Artificial Neural Networks

Tahkola, Mikko; Keränen, Janne; Sedov, Denis; Farzam Far, Mehrnaz; Kortelainen, Juha

Published in:
IEEE Access

DOI:
10.1109/ACCESS.2020.3042834

Published: 07/12/2020

Document Version
Publisher's final version

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 17, 2020, accepted December 1, 2020, date of publication December 7, 2020,
date of current version December 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042834

Surrogate Modeling of Electrical Machine Torque Using Artificial Neural Networks

MIKKO TAHKOLA[1], JANNE KERÄNEN[1], DENIS SEDOV[2], MEHRNAZ FARZAM FAR[1], AND JUHA KORTELAINEN[1]
[1]VTT Technical Research Centre of Finland Ltd., FI-02044 VTT, Finland
[2]Department of Computer Science, Aalto University, FI-00076 Aalto, Finland

Corresponding author: Mikko Tahkola (mikko.tahkola@vtt.fi)

This work was supported by the Arrowhead Tools Project, funded by the European Commission through the European H2020 Research and Innovation Programme, ECSEL Joint Undertaking, and National Funding Authorities from the 18 involved countries under Grant 826452.

ABSTRACT Machine learning and artificial neural networks have shown to be applicable in modeling and simulation of complex physical phenomena as well as creating surrogate models trained with physics-based simulation data for numerous applications that require good computational performance. In this article, we review widely the surrogate modeling concept and its applications in the electrical machine context. We present comprehensively a workflow for developing data-driven surrogate models including data generation with physics-based simulation and design of experiments, preprocessing of training data, and training and testing of the surrogates. We compare neural networks and gradient boosting decision trees in modeling and simulation of torque behavior of a permanent magnet synchronous machine together with selected design of experiments approaches with respect to surrogate accuracy and computational efficiency. In addition, an approach to utilizing domain knowledge to create a hybrid surrogate model in order to improve the surrogate accuracy is shown. The accuracy of the selected hybrid neural network was better than with the gradient boosting approach and was close to the finite element simulation, whereas its run-time efficiency was significantly better compared to the finite element simulation with a speed-up factor of over 2,000. In addition, combining the sampling methods provided better results than the selected methods alone.

INDEX TERMS Artificial neural networks, design of experiments, electromagnetic modeling, machine learning, numerical simulation, permanent magnet machine, surrogate model.

I. INTRODUCTION

Electrical machines (EMs) are an essential part of our everyday life with their applications ranging from small domestic appliances to industrial power plants. Each application requires a particular electrical machine with specific characteristics, and the process of designing such a machine can be performed with the help of mathematical modeling. In addition to design, these mathematical models can be used in condition monitoring, fault diagnosis, system control, and performance evaluation. The ongoing development of the digital twin concept and its enabling technologies, such as data analysis and the Internet of Things (IoT), are also emphasizing the application of computational models.

The physics-based mathematical methods used for low-frequency electromagnetic modeling can be classified into two groups, analytical methods (e.g., electrical equivalent circuit model and magnetic equivalent circuit analysis, MEC) and numerical methods (e.g., finite element (FE) method, FEM). However, each modeling method has its drawbacks. Typically, all methods make a compromise between the computational efficiency and accuracy; a method can be fast but inaccurate, as in the case of various analytical methods, or accurate but computationally expensive, such as FEM. Thus, there is a great need for approaches that are both accurate and computationally inexpensive.

Of various numerical methods, FEM is perceived to be the most workable in the field of electromagnetic analysis of EMs, and has achieved dominance. The FE-based approach for simulating the operation of an EM has the advantage of being able to accurately predict the characteristics and performance of the machine, even without requiring measurements for defining the model parameters. Due to this, it is highly suitable for research, development, and design

The associate editor coordinating the review of this manuscript and approving it for publication was Jinquan Xu.

purposes. Nevertheless, the approach requires data about the geometry and material properties of the machine and its components. In the FE-based approach, the computational domain (i.e., the volume of the main components of the machine) is subdivided into smaller domains called finite elements. In the case of dynamic simulation, the simulated time is also discretized into time steps based on the time scale of the phenomena involved. The variation of the electromagnetic field in an EM is a very fast phenomenon, thus small time steps are required to obtain accurate solutions for field equations. Solving a problem with a large number of time steps and at a high level of accuracy is usually demanding in terms of computational time, making the use of FEM challenging in EM design optimization and system level simulation, where an EM is included as a component. Furthermore, it makes employing FEM directly in digital twins, as well as the control and condition monitoring of machines rather impractical if real-time performance is required.

One solution for the balance between computational accuracy and efficiency are approximative models based on the data from high-fidelity physics-based models. These approximative models are often called surrogate, response surface, emulator, or meta-models [1]. Surrogate models can be classified into three main categories: data-driven, projection-based and hierarchical models [2]–[5]. Hierarchical models are still physics-based models, but with lower fidelity and reduced computational cost compared to an original high-fidelity physical model. In the EM case, data-driven and projection-based surrogates can be constructed based on pre-computed FE simulation results. FEM behaves as a black-box producing data for data-driven models, whereas projection-based models require the extraction of the FEM matrices.

Data-driven regression methods include artificial and deep neural networks (ANNs and DNNs), support vector regression, radial basis functions, kriging, linear and polynomial regression [1], and gradient boosting decision trees (GBDT) [6]. Despite the method, the core of data-driven modeling is to identify relationships between input and output variables from data without knowledge of the system in question. We demonstrate how a reasonably accurate but still computationally lightweight data-driven surrogate model can be generated using machine learning (ML) with FE simulated data.

ML and ANN are an interesting set of technologies widely used for modeling the function and behavior of a system. For example, they provide a unified approach to creating fast computing surrogate models of data that represents the behavior of a system. The approach involves defining model inputs and outputs, finding optimal architecture and learning parameters for the ANN via hyperparameter optimization, and testing the found model with testing data. When combined with simulation, i.e., producing the data for ANN development with computer simulation, the overall process can be mainly automated. One of the drawbacks of the approach is that it usually requires large amounts of data, which means computing numerous simulation cases with a high-fidelity model. In cases when the computing time and the effort needed for generating the data are not an issue, the approach can enable significant run-time performance improvements compared to physics-based simulations.

From the ML perspective, EM modeling can be considered within the scope of two tasks: classification (condition monitoring and fault diagnosis), which requires classification of the current state of the machine into healthy and faulty, and regression (performance evaluation), which tackles the problem of predicting certain characteristics of the machine, such as torque and flux linkages for given initial parameters. In our case, we model a permanent magnet synchronous machine (PMSM) and consider static regression models for the surrogate modeling of PMSM torque behavior. As projection-based surrogates typically model the torque indirectly through a magnetic flux solution and they would need FEM matrices, we concentrate on data-driven options in the experimental part of this study.

The novelty of the work is in providing extensive review on surrogate modeling and its applications in the EM domain, and in demonstrating in detail how machine learning can be employed in surrogate modeling. The main contributions of this article are:

1) To extensively review surrogate modeling and its existing applications in the context of EMs.
2) To thoroughly describe the surrogate modeling workflow from FEM-based data generation to surrogate model training and testing.
3) To compare ANN and GBDT surrogate model accuracy and run-time performance.
4) To compare selected data sampling approaches and their effect on the accuracy of the ANN surrogate models.
5) To evaluate a hybrid model that employs both the physics-based and the ANN approach, and to compare this with a non-hybrid ANN model.
6) To evaluate surrogate model development time and computational run-time efficiency.

The article is organized as follows. In Section II, we present a literature review as background, including FE modeling of EMs, different FE-based surrogate modeling methods with the focus on projection-based, and data-driven methods. In addition, ANNs are introduced and applications of surrogates for electrical machines are reviewed. Section III focuses on our models for the selected application, PMSMs, mainly including FE and ML models and data sampling to generate an ML surrogate. The data preparation and surrogate model development workflows are presented also in Section III. Section IV presents our numerical results, focusing on comparing ANN and GBDT surrogate model types and comparing different data sampling methods in terms of accuracy and computational efficiency of the surrogate models. Finally, Section V discusses the findings and the conclusions of the study.

II. BACKGROUND OF FE-BASED SURROGATE MODELS

Here, we concentrate on different versions of surrogate models of electrical machines, namely the above-mentioned hierarchical, projection-based, and data-driven methods. Since in EM applications these models require FE models or FE simulation data at some level, we begin with multi-fidelity FE models. However, the main focus is on projection-based and data-driven models, and their background and applications concerning EMs.

A. FINITE ELEMENT MODELS AND HIERARCHICAL SURROGATES OF ELECTRICAL MACHINES

Finite element models can be divided into many subcategories based on the model complexity. There are multiple major design choices where one can utilize models with different levels of accuracy, making FEM in its own right a multi-fidelity model [7]. Probably the most influencing design choice is the model dimension, which in the EM case is the decision between a two-dimensional (2D) and three-dimensional (3D) model. The 3D model often requires more than 10-fold higher computational effort, making 2D still the de-facto standard in EM electromagnetic simulations [8], [9]. The second, highly fidelity-influential design choice is the temporal model: static, harmonic, or time-dependent. The rotation and intrinsic nonlinearity of materials often necessitates time-dependent simulation, in which the electromagnetic problem is solved in discrete time occasions called time steps. Its solution produces the most accurate results, but their computational burden is high, due to the large number of time steps often needed. A similar design choice is also to decide whether or not to model eddy currents in massive parts and thin conductors.

A hierarchical (or multi-fidelity) surrogate model is created by simplifying the representation of the physics-based model, i.e., by ignoring certain model aspects or reducing the numerical resolution [5]. In EM modeling, this means that a low-fidelity and computationally less expensive FE model itself is a hierarchical surrogate model to replace the high-fidelity model. The fidelity is reduced, e.g. by the above-mentioned accuracy-related FEM design choices (i.e., by neglecting some physical phenomena), by coarsening the FE mesh or time step density, or by raising the solution residual convergence tolerance. Producing a hierarchical EM surrogate model is fast but as it is still an FE model, and it faces challenges to be fast and accurate at the same time. Intrinsically, the accuracy of a hierarchical surrogate is worse the faster the model is to run. As we emphasize the run-time performance and accuracy, we look for better alternatives.

B. PROJECTION-BASED METHODS

Projection-based methods reduce the complexity of solving the system of equations in numerical simulations by reducing the order of equations. These methods were initially originated in the field of computational fluid mechanics [10] in 1967. Gradually, projection-based models found applications in a wide range of scientific fields from electronics and structural mechanics to biological systems [11]–[13]. In projection-based methods, instead of solving the high-order mathematical equation in the original space, the equation is projected onto a lower dimensional subspace, which is obtained from the span of a set of orthonormal reduced bases, and the problem is then solved in this low-dimensional subspace. In practice, this method reduces the computational complexity by lowering the number of unknown variables. Examples of projection-based methods include the Arnoldi-Lanczos method [14], the reduced basis method [15], the proper orthogonal decomposition (POD) method [16], the proper generalized decomposition method [17], and the a-priori hyper-reduction method [18]. Recently, the application of projection-based surrogate methods in electromagnetic devices has been of interest to a large number of researchers in the field. For example, POD and Arnoldi-based Krylov methods have been used to efficiently reduce the order of linear electromagnetic problems [19]–[21].

Building a projection-based surrogate for nonlinear problems is more complicated than for linear problems. In nonlinear cases, the nonlinear system equations should be solved using iterative methods, which cause extra modeling effort [22]. Various methods have been suggested to reduce the complexity by reducing the order of nonlinear electromagnetic problems, such as subdomain reduction [23], or combining a linear projection-based approach with a trajectory piecewise linear method [24], or with a (discrete) empirical interpolation method [25]. These methods reduce the order of nonlinear problems successfully, but the need for using iterative methods still exists. Although the required number of iterations for solving a nonlinear projection-based model is significantly lower than for solving the full order model, applying an iterative method in real-time applications, e.g. real-time control of an EM, can be still challenging due to very limited computational time. An orthogonal interpolation method to tackle the problem for real-time application has been proposed [26].

The projection-based models mimic the solution process of FEM and are intrusive from the FEM perspective as they are built on the manipulation of the FEM matrices. Hence, they are best in estimating the machine's vector potential and magnetic flux, whereas the torque, for example, is a secondary output solved from the air gap magnetic flux. In a data-driven surrogate model, the torque can be the main output and it is not necessary to estimate the flux. If the main interest lies in global variables such as torque, loss or energy, a data-driven surrogate model can have more freedom as no intermediate results or FEM intrusion are needed. As our aim is to have a surrogate model for PMSM torque modeling, a data-driven surrogate model appears to be a highly interesting and feasible option for us.

C. DATA-DRIVEN SURROGATE MODELS

In data-driven modeling methods, a system is considered as a black-box and a mathematical model is estimated from the

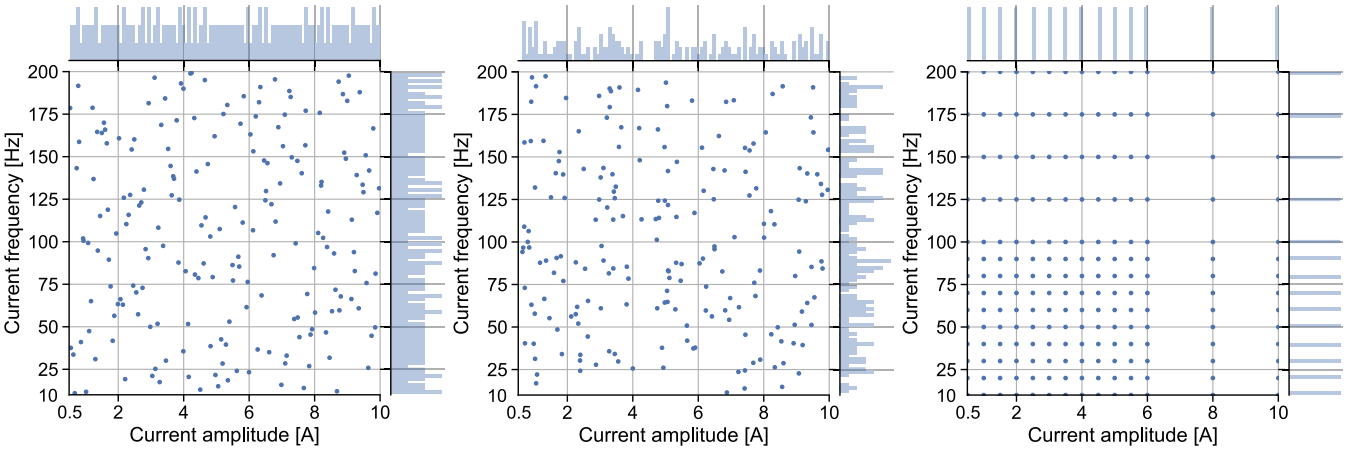

FIGURE 1. (a) LHS, (b) random uniform, and (c) grid sampling methods with two factors and 196 samples. The histograms shown that the samples generated with LHS cover the input space more equally than samples drawn randomly from a uniform distribution, whereas both randomized sampling methods cover the input space better than grid sampling.

relation between the input x_i and y_i output of the system. Therefore, access to the coefficients of the system equations and the internal specifications of the model are not required. A data-driven surrogate model is a simple and computationally inexpensive approximation of a complex model which is created using data produced with the original model. The process of building a surrogate model consists of the following steps: generation of data which include samples for variables x_i and y_i, utilizing data-driven methods to build the model, and verification of the acquired model's accuracy.

The data generation for surrogate modeling can be made with non-adaptive or adaptive Design of Experiments (DoE) techniques. In the field of machine learning, the adaptive methods are better known as active learning methods [27]. In the non-adaptive methods, the whole dataset, i.e., the input and output data, is generated in one go. In adaptive data generation, first a smaller amount of data is generated, and according to the evaluation of the surrogate trained on that data, more data is generated in areas where the accuracy of the surrogate is low [1]. Thus, the total number of simulations to produce data with a computationally heavy model is potentially lower than with non-adaptive sampling. Traditional and widely used non-adaptive DoEs include factorial designs such as full factorial and fractional factorial design, and response surface designs such as Box-Behnken and central composite designs [28]. Space-filling DoEs are another class of techniques that include Latin hypercube sampling (LHS) and Sobol sequences, for example, which aim to cover the input parameter space uniformly [28]. The space-filling characteristic of DoEs are different and may affect the accuracy of the surrogate model. Fig. 1 presents examples of LHS, random uniform, and grid sampling with histograms; showing how the samples cover the parameter space. These examples show that the samples in the LHS method cover the input parameter space more equally than the samples obtained with random uniform sampling. If the system would be highly nonlinear, e.g., within a current amplitude range of 6 to 8 A, the grid sampling method might not capture the dynamics within that range correctly.

There are plenty of models that can be used to build a surrogate, such as polynomial functions, kriging models, radial basis functions, and ANNs [1]. Among all these methods, ANN has become increasingly popular due to its ability to model any complex function given enough training data. In addition to that, having a deep network structure (i.e., many hidden layers) does not necessarily require any laborious feature selection and can work with raw data [29]. FE-based ANN models have been utilized for predicting stress distribution in a 3D printing process [30], bend angles in laser-guided bending [31], and performance of a thermoelectric generator [32], for example.

A gradient boosting decision tree (GBDT) is another widely-used ML model type that is based on combining multiple decision trees, so called base-learners, which are created in series and connected sequentially [33]. Decision trees learn to predict the output values by forming simple if-then decision rules from data. The tree structure includes a root (i.e., input) node, internal decision nodes, and terminal (i.e., leave) nodes that define the possible output values [6]. Other models than decision trees can be employed as the base-learners in the gradient boosting model, such as linear regression and radial basis function models [6].

Depending on the problem under study, the data-driven methods can be categorized into different subcategories such as parametric and non-parametric methods [34], or global and local methods [35]. Global data-driven modeling of a highly complex system can be challenging, for which a network of local models might offer a solution [36]. Global surrogates aim to capture the behavior of a system in the whole input domain, whereas a local model is built to cover smaller parts of it [36]. In addition, a local model network (LMN) approach can be employed. In the LMN approach, multiple local

models form a network in which each model is constructed to approximate a specific area in the input parameter domain. The validity of each local model in the LMN within the input space can be defined by a membership function (also referred to as weighting or validity function). This way the transitions from the input space of one local model to another are smooth [36].

Knowledge of the physics behind the system behavior can be employed by creating a hybrid model. The combination of physics-based and data-driven models can be seen as a hybrid model [37]. A simple way to build one is to utilize a simplified analytical physics-based model to approximate the outputs as functions of the inputs, and train the ML-based data-driven surrogate model with the difference between the predictions of the simple model with the real outputs. The predicted output signal values can be then converted back to originals as a post-processing step.

D. ARTIFICIAL NEURAL NETWORKS

Even though numerous model types for data-driven modeling exist, the focus in this study is on ANNs which mimic the human brain's way of processing information. In the experimental part, we also compare the ANN and GBDT performance of the models. The most basic ANN architecture is a multilayer perceptron (MLP) as shown in Fig. 2. MLP is a feedforward network as there are no recurrent connections, neither from the output nor from hidden layers to previous layers. The first layer of the network is the input layer, which distributes the input values to each neuron in the first hidden layer. A hidden neuron can be thought of as a simple processing unit, as shown in Fig. 2. The input values coming from the neurons of the previous layer has its own weights, and each hidden neuron has its own bias parameter. The weighted inputs and the bias are added up and the sum is fed for the input to activation function, which can be linear or nonlinear. The output of the activation function is then distributed to the next layers, or if the neuron is in the output layer, it is the model output.

Supervised learning is a class of ML, in which the expected output values are known and the model is updated during the learning process to minimize selected error metric between the predicted output and the ground truth output values. ANN training in a supervised manner means that the weights and biases are adjusted so that the model fits the training data. The generalization ability of an ANN is highly important. If the ANN weights are fitted too well on the training data, the model is overfitted and the accuracy is worse with slightly different data. ANN training usually involves hyperparameter optimization (HPO), which is done, for example, to find a structure for the network and adjust other hyperparameters such as the learning rate or the weight initialization method, which have an effect on the learning. There are different HPO methods, such as grid search and random search, in which the hyperparameter sets are independent [39]. HPO methods such as genetic algorithms, particle swarm optimization and

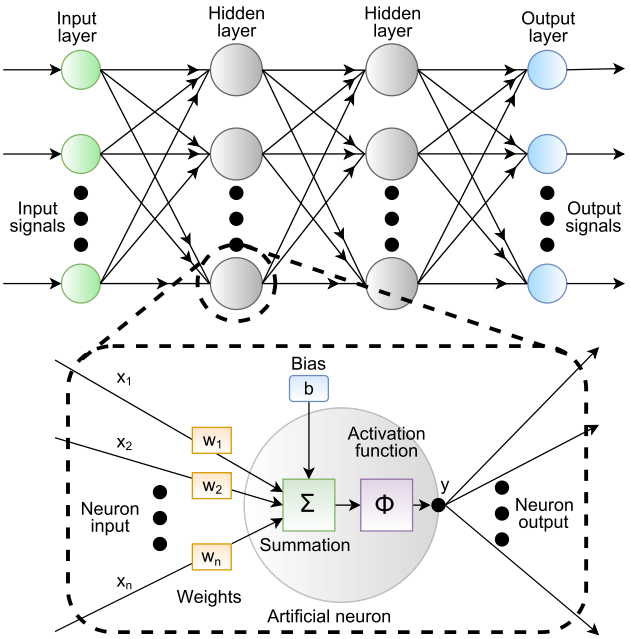

FIGURE 2. Structures of an MLP network and artificial neurons. Adapted from [38].

Bayesian optimization make use of previous training results as the optimization progresses [40].

Various methods can be used to evaluate the generalization ability of the ANNs. Available data is usually divided into three parts – training, validation, and testing data. Validation data is used to evaluate the generalization ability of models trained with different hyperparameters. Supervised learning processes, including ANN training, can be monitored by comparing the training and validation error as the training progresses. If the validation error starts to increase at some point of the training, while the training error is decreasing, it implicates overfitting. A method called early stopping can be used to prevent overfitting. Early stopping can be applied to stop the training when the validation loss starts to increase or when it has not decreased during N previous training iterations [41]. The best model from the hyperparameter optimization is selected by comparing the validation errors. Finally, the selected model accuracy is tested on an independent test dataset that has not been used in training or validation. This is called the holdout method, which is the simplest type of cross-validation technique [38]. The test dataset is required because estimation of the model generalization with the validation dataset becomes unreliable when it is used in hyperparameter optimization to select the optimal set of hyperparameters. Multifold (i.e., k-fold) cross-validation is another well-known method, in which the training data is divided into k subsets and the same model is trained k times so that each of the subsets has been used in validation, whereas other subsets form the training data [38]. Development of a data-driven model may also utilize feature engineering, including feature extraction and selection. Examples of feature extraction are variable transformations and the generation of polynomial features from the original features,

i.e. variables. Input feature selection techniques include wrapper, embedded and filter methods [42].

E. APPLICATIONS OF SURROGATES IN ELECTRICAL MACHINES

The application of surrogate techniques in electromagnetic devices has been of interest to a large number of researchers in the field recently. The applications include design optimization, fault diagnosis and condition monitoring, and control of such devices. In addition, projection-based surrogates have been utilized for uncertainty quantification [20].

1) DESIGN OPTIMIZATION

Surrogate model-based optimization (SMBO) methods have been widely used in design optimization. The methods can employ a global or a local surrogate, or both [43], [44]. A global surrogate model is built to cover the entire design domain, whereas a local one approximates a smaller area of the domain. Due to this, a local approach can utilize less complex models and require less data than a global one [43]. Alternatively, these two approaches can be combined. In that case, a global surrogate can be utilized first to explore the entire design space and find the most promising areas where local models are then built to search for a local optimum [45].

SMBO has been applied with FEM to accelerate optimization tasks, e.g., in [46], where the authors present a method for optimizing a doubly-fed induction generator winding design to maximize the power yield. Design of an interior PMSM was optimized with a surrogate-assisted multi-objective optimization algorithm in [47]. Giurgea *et al.* [48] applied surrogate modeling in design optimization for a PMSM. The authors in [49] compared the accuracy of surrogate models that were created using FE simulation data produced with different design of experiment strategies. Surrogates and genetic algorithms have been combined to find an EM design that produce optimum constant power speed range [50], [51], for example, and to optimize the weight of an EM [52]. Different design of experiment strategies were employed in data FE generation in [53] to optimize brushless direct current motor design. Further, the torque performance of such motor was optimized in [54].

2) FAULT DIAGNOSIS AND CONDITION MONITORING

ML has been employed for anomaly detection and machine condition monitoring. Anomaly detection is easier to carry out as only data from normal operational conditions is needed. However, data from faulty operational conditions is required to classify the faults or machine condition. Janssens *et al.* [55] have developed a convolutional neural network (CNN) -based method to automatically learn features from vibration data, which characterize bearing faults. Multiple ML methods in electrical motor fault diagnosis are presented and compared in [56]. Wen *et al.* [57] propose a CNN-based approach to convert time-domain measurement signals into two-dimensional images, from which relevant features are then extracted. Senanayaka *et al.* [58] present a CNN-based online fault diagnosis system. The classification is based on statistical features extracted from handled signals, and principal component analysis is utilized to reduce the number of features to reduce the model complexity and enhance model generalization. Quiroz *et al.* in [59] utilize a method based on random forests to diagnose broken rotor bar failure in a line start-permanent magnet synchronous motor. In [60], the authors propose a CNN-based method, called dislocated time series CNN (DTS-CNN), to classify faults from raw signals. Jia *et al.* [61] present a DNN for fault classification, an approach which removes the need for manual feature extraction and variable selection.

3) CONTROL OF ELECTRICAL MACHINES

In the control of EMs, a model of the machine is used to represent the machine. This model is normally analytical, or it is based on one or a few lookup tables [62]. Due to the potential of surrogate models in estimating the machine behavior accurately, the application of surrogate models in control of electrical machines has recently attracted increasing attention. For example, numerous publications have presented the efficiency of model predictive control (MPC) in real-time control of PMSMs [63]–[65]. However, only a few published articles are available that suggest the utilization of an FE-based surrogate model in these applications, even though there are articles presenting the utilization of non-FE-based ANNs for PMSM control [66]–[68]. The main challenge in this utilization is the computational time constraints of the real-time applications. This means that the surrogate should not only present the original model precisely, but also should have the capacity to compute the solution quickly enough.

Pinto *et al.* [69] have developed a dq0 flux-linkage-based model for a PMSM, using FE calculations for various dq0 currents at different rotor positions. The torque and qd0 currents resulting from the proposed model are similar to the ones obtained with FEM, while the simulation time of the proposed model is significantly reduced. Thus, the authors have proposed this model for controller design and hardware-in-the-loop (HIL) simulation. Drobnic *et al.* [70] propose a fast flux-linkage model (FLM) of a nonlinear interior permanent magnet synchronous machine (IPMSM), which is parameterized with a set of data calculated by FEM. The proposed FLM is compared with a current model (CM), driven also from the FEM. Since the proposed FLM is 20% faster than the CM, the authors suggest the FLM for computationally intensive application with an excessive real-time time span. Farzam Far *et al.* [26] have developed a projection-based model of an IPMSM that has the stator currents in the rotor frame of a reference as inputs and the nodal values of the magnetic vector potential, and thereafter the flux linkages, as outputs. This model is implemented in an embedded processor of a corresponding drive to control the machine prototype in real time.

III. DATA-DRIVEN SURROGATE MODELING OF PERMANENT MAGNET SYNCHRONOUS MACHINES

The section discusses the selected application, namely PMSM modeling. Popular physics-based models used with PMSMs are first discussed as a background to show which kinds of models the surrogates could replace. However, the main usage of FE models here is for producing data for creating surrogate models of PMSMs. Thus, we focus on the workflow of the data generation and data preparation processes. Finally, we present the surrogate model development including hyperparameter optimization.

A. PHYSICS-BASED MODELS OF PERMANENT MAGNET SYNCHRONOUS MACHINES

The development of rare-earth magnetic materials has resulted in the advancement of PMSMs. Rahman [71] summarizes the history of this development. A PMSM is a synchronous machine that consists of three (or more) phase windings in a stator and permanent magnets (PMs) in a rotor for the field excitation. Fig. 3 represent the main elements of a typical PMSM (the geometry represents the test case examined in Section IV).

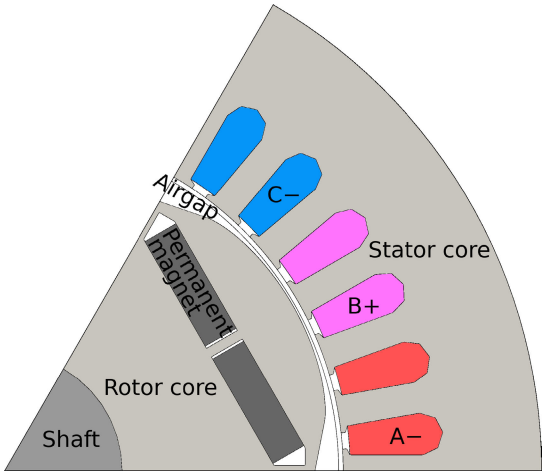

FIGURE 3. A cross-section of an IPMSM, showing only 1/6 of the geometry by symmetry. The main elements of the machine are shown where A−, B+ and C− are the coil sides of the three phases of the stator windings.

Due to the presence of PMs, PMSMs can produce torque at zero speed and have higher efficiency and torque per unit volume compared to induction machines. Therefore, PMSMs are suitable for applications in which high-performance and high-efficiency machine drives are required, such as in wind power generation, electrical vehicles, and robotics. Depending on the mounting locations of the PMs, on the surface or inside the rotor, PMSMs are categorized as surface PMSMs (SPMSMs) or interior PMSMs (IPMSMs, see Fig. 3), respectively. Compared to IPMSMs, SPMSMs produces a higher air gap flux density, but the mechanical robustness and the ratio between the quadrature and direct-axes inductances are lower. Thus, unlike IPMSMs, SPMSMs are mainly used in low speed applications.

PMSMs can be modeled at different levels of computational accuracy and efficiency. One typical analytical model of EMs is based on electrical equivalent circuit equations and describing the phase quantities in a d-q rotor reference frame. The voltage equations defining the electrical dynamics of the PMSM in d-q frame can be presented as

$$V_\mathrm{d} = R_\mathrm{s} i_\mathrm{d} + \frac{d\lambda_\mathrm{d}}{dt} - \omega_\mathrm{e}\lambda_\mathrm{q}, \tag{1}$$

$$V_\mathrm{q} = R_\mathrm{s} i_\mathrm{q} + \frac{d\lambda_\mathrm{q}}{dt} + \omega_\mathrm{e}\lambda_\mathrm{d}, \tag{2}$$

$$\lambda_\mathrm{q} = L_\mathrm{q} i_\mathrm{q}, \tag{3}$$

$$\lambda_\mathrm{d} = L_\mathrm{d} i_\mathrm{d} + \lambda_\mathrm{m}, \tag{4}$$

where $(V_\mathrm{d},\ V_\mathrm{q})$, $(i_\mathrm{d},\ i_\mathrm{q})$, $(\lambda_\mathrm{d},\ \lambda_\mathrm{q})$, and $(L_\mathrm{d},\ L_\mathrm{q})$ are the d- and q-axes voltages, currents, flux linkages, and inductances, respectively. R_s is the stator resistance, ω_e is the electrical angular speed of the rotor and λ_m is the PMs flux linkage. The inductances, resistance and flux components can be identified by analytical equations or alternatively from FE results. The electromagnetic torque can also be defined by d-q parameters as

$$T_\mathrm{e} = \frac{3p}{2}\left[\lambda_\mathrm{m} i_\mathrm{q} + (L_\mathrm{d} - L_\mathrm{q}) i_\mathrm{d} i_\mathrm{q}\right], \tag{5}$$

where p is the number of pole pairs in the machine.

The aforementioned electrical equivalent circuit equations are the most simplified mathematical model that is used to present a PMSM. The most significant disadvantage of this model is that all parameters need to vary sinusoidally in the coordinates. Hence, this limits the accuracy to simulate the behavior of an actual machine that consists of nonlinear materials (even if saturable models exists, e.g. [72]). One more advanced and accurate model class are the MEC models, which are based on reluctance networks [73], [74]. An MEC models the machine's cross-section with geometry-dependent reluctances. The reluctances can be in parallel or in series, producing a magnetic circuit that models the EM behavior. Additionally, material nonlinearities can be included in the model. MEC is a multi-fidelity model, and its accuracy depends on features taken into account in the magnetic circuits. In its simplest form, it is an analytical model, but by adding more circuit parameters, it can be interpreted as a numerical method, and its accuracy draws close to that of FE analysis. However, it is computationally faster than FEM. One disadvantage of MEC is that you need a preconceived idea of the flux paths, making it in some cases difficult to automate the reluctance network generation compared to model generation and meshing with FEM. Further, the eddy currents cannot be easily included in MEC without significantly increasing the model complexity.

For generating data for an FE-based surrogate, an accurate model of the EM with nonlinear material is required, and even more significantly a method for which model generation

is easy to automate. As mentioned previously, FEM is an efficient tool for such a purpose.

Let us consider a current-fed PMSM with a 2D time-dependent solution with FEM. In this case, the solution to the field can be expressed by a magnetic vector potential A (with only a z-component, i.e. $A = A_z e_z$, where A_z is the z-component and e_z is the z-directional unit vector) and this vector potential is used to compute all the quantities of the machine such as the distribution of the flux density, field strength, voltages, currents, torque, and flux linkages. The vector potential A and magnetic flux density B can be solved from

$$-\mathrm{div}\left(\frac{1}{\mu}\,\mathrm{grad}\,A_z\right) + \sigma\,\frac{\partial A_z}{\partial t} = J_\mathrm{s} + \mathrm{curl}\,M, \tag{6}$$

$$B = \mathrm{curl}\,A, \tag{7}$$

where μ is the magnetic permeability, σ is the electrical conductivity, J_s is the source current density, and M magnetization of the permanent magnets. Discretizing the weak format of the field equation and using FEM, we can present the problem with an algebraic system of equations as

$$\mathbf{K}\,\boldsymbol{a} + \mathbf{M}\,\dot{\boldsymbol{a}} = \boldsymbol{f}, \tag{8}$$

where $\mathbf{K}$ and $\mathbf{M}$ are known as the stiffness and mass matrices, respectively. $\boldsymbol{a}$ consists of the nodal values of the magnetic vector potentials and $\dot{\boldsymbol{a}}$ is the time derivative of $\boldsymbol{a}$. $\boldsymbol{f}$ is the source vector resulting from the input current and the curl of the magnetization produced by the PM. (8) can be solved by various methods, such as the Euler method, the Runge-Kutta method, or the Gear method. We choose to use a backward Euler approach, where after discretizing the problem in time, the nodal values at time step t_k can be solved from:

$$\left(\mathbf{K} + \frac{1}{\Delta t}\,\mathbf{M}\right)\boldsymbol{a}^k = \frac{1}{\Delta t}\,\mathbf{M}\,\boldsymbol{a}^{k-1} + \boldsymbol{f}^k. \tag{9}$$

In nonlinear problems, since the stiffness matrix $\mathbf{K}$ depends on the magnetic nodal values, an iterative method is required to solve (9).

As mentioned previously, knowing the nodal values of the potential, one can compute the magnetic flux density B by (7), and the torque T acting on the rotor of the machine with Maxwell stress tensor as

$$T = \frac{1}{\mu_0\,(r_\mathrm{s} - r_\mathrm{r})}\int_{\mathrm{S_{ag}}} r\,B_\mathrm{r}\,B_\phi\,\mathrm{dS}, \tag{10}$$

where r_s and r_r are the outer and inner radii of the air gap,[1] respectively. $\mathrm{S_{ag}}$ is the cross sectional area of the air gap. B_r and B_ϕ are the radial and tangential components of the flux density [75].

The order of the finite elements and the mesh size determine the accuracy of the machine quantities and higher-order finite elements or a finer mesh are required to improve the accuracy. This typically leads to a large set of equations, which in return increases the computational cost significantly. In this paper, we propose a surrogate model to reduce this computational cost.

In Section I, we mentioned control and system level models as possible applications of surrogates. At present, the most popular models in those applications are d-q equivalent circuit equations (described above) and lookup tables (LUTs) [62], [76]–[78]. In this usage, LUT typically means a multi-dimensional table of FE pre-computed values with selected sets of input parameters and means to linearly interpolate the output for a new input value. LUT values can be losses, torques, inductances, flux linkages, as a function of d- and q-axis current and rotor angle. Roughly, LUT models are typically faster than FEM or even MEC models, and their accuracy is innately lower compared to FEM, but in the vicinity of MECs. However, LUT requires a considerable amount FEM pre-computation, like the data-driven surrogates. We propose a surrogate to keep the run-time computational efficiency close to LUTs, but with better accuracy.

B. DATA GENERATION WITH PHYSICS-BASED FE MODEL

A simulation model of the IPMSM in Fig. 3 was used to produce training data for ML. The IPMSM was modeled with an open-source FEM software, Elmer [79], by ignoring the losses and eddy currents in the machine and using current-fed 2D time-dependent solution. The Elmer software numerical solution process is described in [80]. In the model, the core material of the electrical machine was modeled with a nonlinear single-valued B-H curve as described in [81]. The parameters of the IPMSM are given in Table 1.

TABLE 1. Parameters of the IPMSM.

Parameter	Value	Parameter	Value
Power	2.2 kW	Airgap length	1 mm
Rated current (rms)	4.3 A	# of stator slots	36
Rated frequency	75 Hz	Stator outer diameter	165 mm
Rated speed	1,500 rpm	Stator inner diameter	104 mm
# of pole pairs	3	Effective axial length	115 mm

The FE mesh (Fig. 4) for Elmer was produced with FreeCAD [82] and GMSH [83], using the FreeCAD Python interface. The mesh has in total 2,268 nodes and 4,177 triangular elements. Since first order elements are used, the mesh in the air gap needs to be very dense to produce reasonably accurate torque results for ML. In a time-dependent simulation, 400 time steps were simulated and the time step length was adjusted to have 200 time steps in an electrical period. Fig. 5 shows the magnetic flux density result of the model just after one mechanical period, with the rated current magnitude and frequency.

Two datasets were produced for data-driven modeling by performing parameter sweeps with the Elmer FE simulation model of the IPMSM using a desktop computer with an Intel Xeon E5-2640 v3 processor. The model was fed with a sinusoidal input current, characterized by a current frequency and amplitude that were varied to produce rich

[1] In case on nonuniform air gap like in Fig. 3, inner radii used in the equation (10) is the maximum value of the geometrical inner radii.

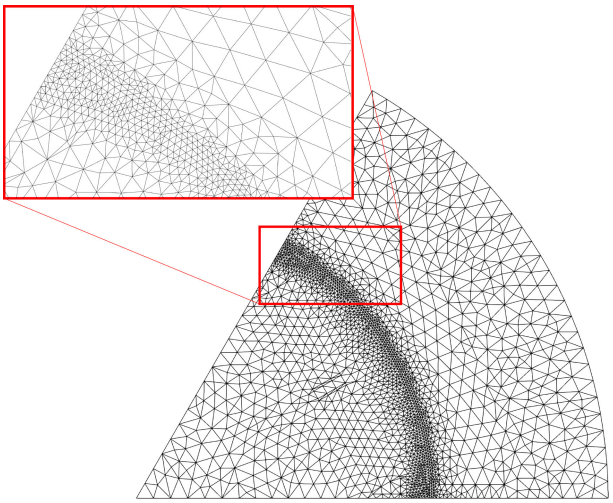

FIGURE 4. Simulation FE mesh of the case, with a zoomed region in the air gap.

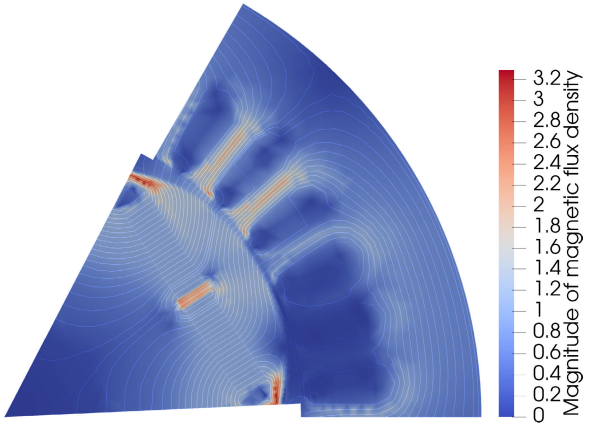

FIGURE 5. Simulation result with flux lines and the magnetic flux density distribution.

datasets. Other machine parameters were constant, e.g. the rotor initial position and the number of time steps were kept same in all parameter sweeps. The actual logged inputs in the datasets are time series of current values in three phases. The logged output is an air gap torque time series computed from Maxwell stress tensor using (10). The first dataset was generated using grid sampling and the second using the Latin hypercube sampling (LHS) method. In this context, choosing the cases is referred to as sampling. As the FE simulation is deterministic, no replications are included in the DoEs.

C. DATA SAMPLING APPROACHES

The grid and LHS datasets consist of in total 196 and 1,000 cases, respectively. Cases in the grid dataset are in a grid form as Fig. 1 (c) shows. To produce a rich set of data, the maximum current magnitude was selected to ensure highly saturated results in the data. The current amplitude values ranges from 0.5 to 10 A and the current frequency values ranges from 10 to 200 Hz, both with 14 different values in the grid. The same ranges were used to generate the LHS dataset. The first 400 time steps of input current (phase A) signals and corresponding output torque responses for six cases from the LHS dataset are shown in Fig. 6.

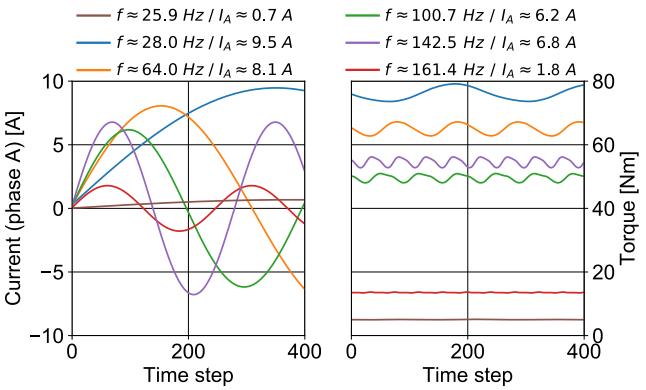

FIGURE 6. An example of FE simulation results, showing one of the three phase current signals (left) and corresponding torque responses (right) of six cases.

The dynamics of the modeled system contain nonlinearities, and the torque behavior of the machine is different at low current amplitudes compared to high ones. In the initial study of hyperparameter space limits, the model error on both the training and validation datasets was higher in cases with a low current amplitude than in those with higher current amplitudes. The physical reason for this is that different terms of torque dominate in different current values. With zero current, all torque is cogging torque (i.e. the torque produced by PMs reacting to the stator teeth) and the average torque is zero. Cogging torque produces torque ripple, a variation of the torque around its average. At low currents, the cogging torque still dominate. At a high current, there is a high average torque due to interaction between the PMs and stator coils. Further, the magnetic core of the machine is so saturated that other torque ripple terms (e.g. due to nonsinusoidal airgap field) dominate the cogging torque. Hence, the torque timeseries waveform changes gradually from 0 A to 10 A, when the saturation-related term starts to dominate the cogging torque.

To ensure unbiased model validation and testing, the input space was divided into five sets by the current amplitude values as shown in Table 2. Cases for validation and testing datasets were selected from the LHS dataset pseudo-randomly so that the same number of cases were drawn from each of the five current amplitude sets. The validation and testing cases were excluded from the LHS dataset before selecting cases for different training datasets. The model validation and testing datasets included 150 and 190 cases, respectively.

The naming of the datasets is shown in Fig. 7. The grid dataset, referred to as g196, was used in the training as it is. Cases for other training datasets were drawn from the remaining set of cases in the LHS dataset to study how much the number of cases affects the model error. Four training datasets (e50, e100, e200 and e300) were created from the LHS dataset similarly to how the validation and testing

TABLE 2. Number of cases in different input current amplitude ranges in the training, validation, and testing datasets.

Dataset	Current amplitude range [A]				
Abbr.	0.5–2	2–4	4–6	6–8	8–10
e50	10	10	10	10	10
e100*	20	20	20	20	20
e200	40	40	40	40	40
e300	60	60	60	60	60
w50	15	12	10	8	5
w100	30	25	20	15	10
w200	60	50	40	30	20
w300	90	75	60	45	30
g196*	42	56	56	14	28
g196+e100*	62	76	76	34	48
Validation	30	30	30	30	30
Testing	38	38	38	38	38

* Three types surrogate models were trained with the e100, g196 and g196+e100. More details in Fig. 7 and III-D.

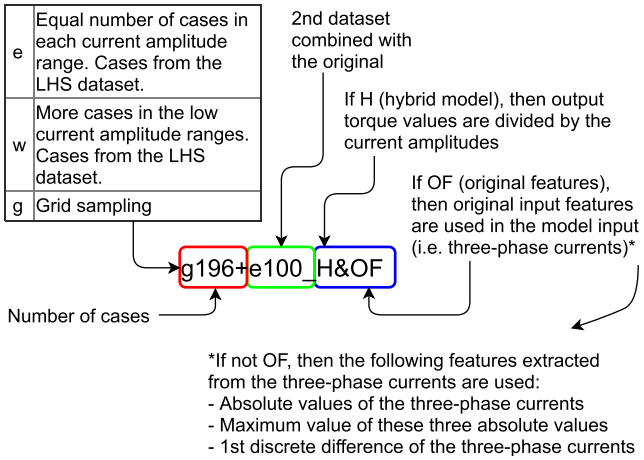

FIGURE 7. Notation used with the training datasets. Green and blue parts are optional.

datasets were formed, i.e. by drawing cases pseudo-randomly from the LHS dataset. Four more datasets (w50, w100, w200 and w300) were created with more cases in the low input current amplitude ranges and less in the high amplitude ranges. This was done in order to study if having more cases in the low current amplitude area of the input space improve the model accuracy. Finally, a dataset consisting of 296 cases (g196+e100) was created by combining the samples of datasets g196 and e100, in order to compare it to the e300, w300 and g196 training datasets. The difference between the g196+e100, e300 and w300 datasets is that in the first one, the grid sampling ensured that there were cases in the outer edges of the input space and additional cases were located in between the grid points in a randomized manner. A comparison between the g196 and g196+e100 datasets was made to see if the randomized points could improve the model accuracy compared to pure grid sampling. The case distributions of the training datasets g196, e200 and w200, and validation and testing datasets across the input space are shown in Fig. 8.

D. TRAINING DATA PREPARATION

Individual cases in the grid and LHS datasets had different sampling frequencies, i.e. different time step lengths.

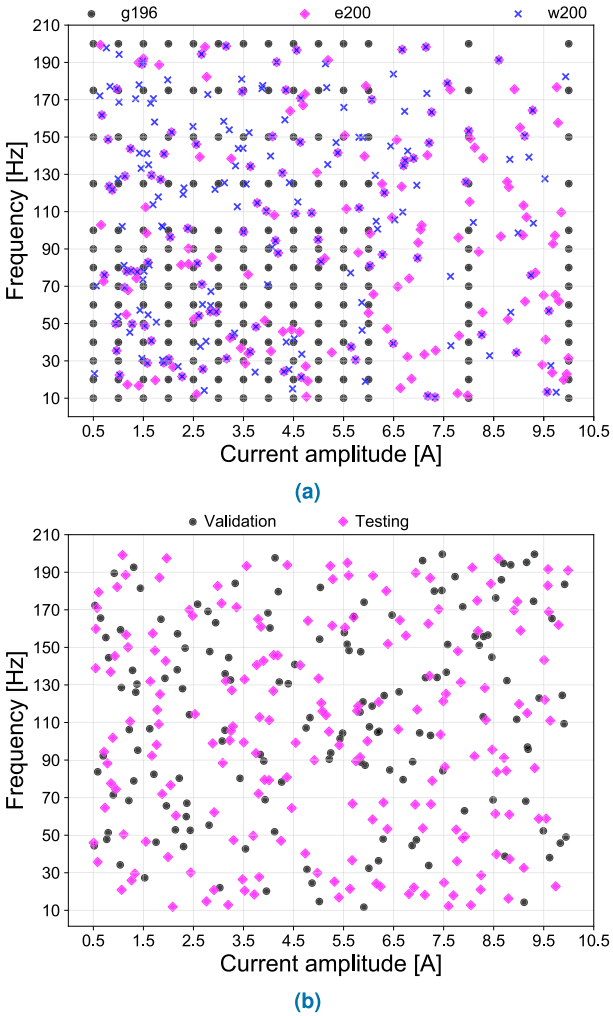

FIGURE 8. Samples in the training datasets g196, e200 and w200 (a), and the validation and testing datasets (b). The subfigure (a) shows the difference between e and w datasets. In the latter there are more samples in the low current amplitudes.

In Elmer, the sampling frequency was selected on the grounds of the input signal frequency. In the low input current frequency cases the sampling frequency was lower than in higher input current frequencies. The number of FE samples[2] in each case was 200. In order to unify the sampling frequency, i.e. to make the time step length equal, cases sampled with lower than the highest frequency were upsampled to match the highest sampling frequency. New samples were generated with linear interpolation. After the upsampling, the lowest frequency (10 Hz) cases included 4,000 samples since the time step length in those cases was 20 times longer than in the highest frequency (200 Hz) cases.

Using merely the input current signal values in the model input did not result in good model accuracy at low current amplitudes. Therefore, seven features were computed from the three input current signals – the absolute values of each

[2]Originally there were 400 samples in each case as the sinusoidal input in the FEM simulation included two full cycles. Since the input and output data of the cycles were copies of each other, the second cycle was removed.

signal, the maximum value of these three absolute valued signals, and the first discrete difference of the three input current signals. The actual values of the three input current signals were excluded from the model input, which resulted in a total of seven input variables. The model output was the torque of the PMSM. The input and output signals were scaled to range from −1 to 1, by scaling with the minimum and maximum values of each signal in the training dataset. However, even after generating the additional input features, the low current amplitude accuracy of the surrogates was not sufficient.

Since the output torque depends approximately linearly on the input current amplitude (see equation (5)), versions of the training datasets e100, g196 and g196+e100 were created, in which the torque values were normalized (divided) by the input current amplitude. These are denoted by H (hybrid) and H&OF (hybrid and original input features). The difference between the two was that in the H versions, the seven extracted features were used in the model input, whereas in H&OF, the original features were used. With the H and H&OF setup, the models learns to predict the normalized values, and these values are converted back to the original scale in post-processing. The assumption was that training a neural network model would be easier when the range of output values was narrow. This kind of model could be seen as a simple hybrid model in which domain knowledge of the physical relationships between input and output variables is utilized. This linear dependence of torque on current is really an approximation and, hence, the ML part of the hybrid model takes care of the more complicated torque terms depending nonlinearly on currents. In these cases, the predicted output values were converted back to the original scale as a post-processing step by multiplying them with the input current amplitude.

E. SURROGATE MODEL DEVELOPMENT

The ANN and GBDT models were trained on a computer with an Intel Xeon E5-2690 v4 processor. The Keras API of Tensorflow version 2.1 [84] was used in the experiments to develop the ANN models. A regression type of GBDT models were developed with the Python API of the LightGBM gradient boosting framework [33]. The ANN and GBDT models built in this study were feedforward models, i.e. there are no recurrent connections. The model input consist of the input feature values in the current time step.

For the ANN, the hyperparameter optimization included training eight models with one or two hidden layers and 128, 256, 512 or 1024 hidden neurons. The networks had Rectified Linear Unit (ReLU) activation functions. The output of a ReLU [29] function is given as

$$f(x) = \max(0, x). \tag{11}$$

Neural network weights were initialized using a Glorot uniform initialization [85]. A first-order gradient-based algorithm called the Adaptive moment algorithm (Adam) [86] was used in the neural network weight optimization with a learning rate of 0.0002. After each training iteration, the batch size was set to increase from the initial value of 32 by

$$\frac{N_{\max} - N_{\text{ini}}}{N_{\text{e}}} = \frac{12800 - 32}{3200} \approx 4.257, \tag{12}$$

where $N_{\max}$ was the maximum batch size, N_{ini} was the initial batch size and N_{e} was the number of epochs. The resulting batch size value was rounded to the nearest integer. Using an adaptive batch size in the ANN training is discussed in [87], [88]. For example, when the batch size is 32 and the number of samples is 3,200, samples of the training dataset are divided into 100 batches. This means that during one epoch of neural network training, the weights of the network are updated 100 times. The number of epochs defines how many times this process is repeated during the training. The error metric that the model parameter optimization algorithm minimizes was mean squared error (MSE) for both ANN and GBDT models.

The maximum number of training epochs was set to 3,000 but the model training was conducted one epoch at a time and the validation error (MSE) was computed after each one. The model was saved when the validation error was lower than the lowest so far achieved validation error. Early stopping was used as a regularization method to avoid overfitting and the tolerance was set to 30 epochs, meaning that if the validation error did not decrease during the number of epochs counted from the currently lowest achieved validation error, the training was stopped. After that, the model with the weights which resulted in the lowest validation error was saved.

The GBDT hyperparameter optimization included 500 combinations of pseudo-randomly chosen hyperparameters chosen within the search space shown in Table 3. Other hyperparameters were left as default values.

TABLE 3. Hyperparameter search space for GBDT.

Hyperparameter	Options
Number of trees	2,000–10,000
Maximum tree depth	−1 (unlimited), 5, 10, 20 or 40
Learning rate	$1e^{-2}$–1
L1 and L2 lambda	0, 0.15, 0.3

The torque values in different cases are different magnitudes in other than the H and H&OF training datasets. As an example, with a current amplitude of 0.5 A, the torque value ranges in the output time series change between 3.7 to 3.8 Nm, whereas with current amplitude of 10.0 A, the values are between 77.5 to 83.4 Nm. This means that an error of 1 Nm in the torque estimation, for example, in the low amplitude case would be relatively high compared to a 1 Nm error in the latter case. Therefore, instead of using MSE as the metric to select the best model from the hyperparameter optimization, a normalized root mean square error (NRMSE) was used. The NRMSE was computed for each case separately, and the average NRMSE of all case NRMSEs was used to choose the best model. From the electrical machine

torque modeling perspective, mainly for the above-mentioned relative error reasons, the maximum value of the NRMSE over dataset is a good measure of the accuracy and usability of the model. In addition, the average NRMSE is a good measure of the average relative model error.

IV. RESULTS OF DATA-DRIVEN SURROGATES

The computational accuracy and efficiency of the ANN and GBDT model types are first compared in this section and further experiments are made with ANN with different sampling approaches to compare grid and multiple randomized strategies. Employing domain knowledge in the model structure to create a hybrid ANN model is presented. Finally, the computational efficiency of ANN surrogates is evaluated and the surrogate model development time including data generation with FEM and training the surrogates is discussed.

A. COMPARISON OF ANN AND GBDT MODELS

ANN and GBDT regression models were trained on the g196+e100_H dataset. The hyperparameter optimization for both was explained in III-E. The ANN hyperparameter optimization was repeated ten times and in each repetition, a different seed value for random number generator was used to start the training from different initial network weights. The final ANN model (2 layers, 512 neurons in both layers) was selected by the lowest average NRMSE of these models. The best GBDT model included about 7,500 trees, each with 46 leaves and a maximum depth of 40. The GBDT model was trained with a learning rate of approximately 5.32^{-2}, and both regularization parameters L1 and L2 lambda were 0.3.

The hyperparameter optimization for GBDT took 123.3 min (wall time), i.e. 14.8 s per trained model on average. The corresponding time for ANN was 342.6 min (wall time) for hyperparameter optimization which is 257 s per model on average (80 models were trained). Even though the hyperparameter optimization for GBDT took less time than for ANN, there was quite a gap in the actual model run-time efficiency. The simulation time of the best GBDT model was 146.8 s for the 190 test cases, resulting in an average of 0.77 s per case. For the ANN model, the simulation time was 12.9 s, i.e. 67.8 ms per case, which makes the ANN model about 11 times faster than the GBDT model.

The ANN model was not only faster in this comparison, but also more accurate. In this comparison, we felt the computational accuracy and efficiency of the model to be more important than the time used to develop the model through hyperparameter optimization. Fig. 9 shows the test cases in which the NRMSE of the GBDT and ANN models trained with g196+e100_H datasets were the worst. The average, minimum, and maximum test NRMSE values of the GBDT model were 2.17%, 0.60% and 13.11%, whereas the corresponding error values for the ANN model were 1.14%, 0.43% and 5.23%. In addition to better accuracy, the ANN model output changes smoothly, whereas the GBDT output is step-wise, which is typical for the model type, and is caused by the model structure. Step-wise signals in EM model applications produce high-frequency errors. That is, a fundamental frequency solution could be usable, but harmonics solutions will probably be distorted. Based on this comparison, we chose to continue with the ANN models in the next experiments.

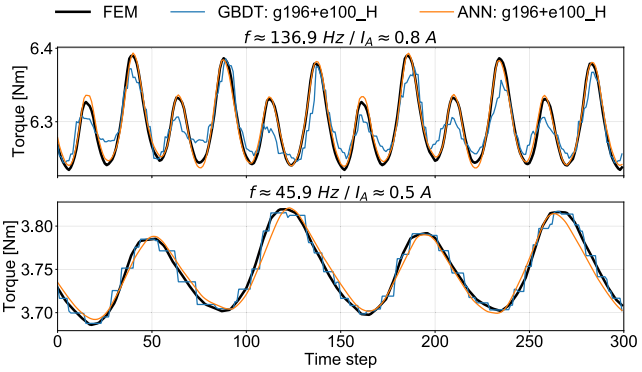

FIGURE 9. Torque estimations of the GBDT and ANN models compared to FE simulation results. The upper plot shows a test case in which the GBDT accuracy is the worst with an NRMSE of 13.1%, whereas for the ANN it is 3.5%. The bottom plot shows the worst test case for ANN with NRMSE of 5.2%, whereas for the GBDT is is 4.8%.

B. INFLUENCE OF DATA SAMPLING ON ANN SURROGATE PERFORMANCE

Data sampling affects the accuracy of data-driven models. Experiments were made to study influence of the selected data sampling approaches and the number of training samples on the ANN surrogate model accuracy. The effect of employing a domain knowledge-based hybrid model is examined with the datasets e100, g196 and g196+e100. In this context, the need for generating additional input features from the original ones is discussed.

1) ARTIFICIAL NEURAL NETWORK STRUCTURE

The selection of the best ANN model size for each training dataset (see Table 2) was carried out utilizing hyperparameter optimization as described in III-E. Again, the hyperparameter optimization procedure was repeated ten times for each training dataset. These results are presented and compared first.

The validation results are shown in Fig. 10. With the networks of one hidden layer, the average NRMSE decreases as the number of hidden neurons increases. Having two hidden layers instead of one but keeping the same number of hidden neurons reduces the average NRMSE as well. The best average accuracy with datasets e50, w50, e100, w100 and w200 was achieved with a network size of 2×1024. For datasets e200, e300 and g196+e100, a network size of 2×512 resulted in the best average NRMSE. The corresponding best sizes with datasets g196 and w300 were 1×512 and 2×256.

The best average NRMSE values with the normalized output training datasets e100_H, g196_H and g196+e100_H were obtained with neural network sizes of 2×256, 2×512 and 2×1024, respectively. The average validation NRMSE values with the hybrid models were worse compared to their non-hybrid counterparts when the network had only one hidden layer, as shown in Fig. 10. However,

FIGURE 10. Average validation NRMSE of models with different sizes and trained on datasets constructed with different methods and number of samples.

when a second hidden layer was added, the corresponding NRMSE values, the hybrid versions perform averagely better than the non-hybrid ones. This suggests that normalizing the torque values with the input current amplitudes enables better learning when the network is big enough. The lowest average validation error with the training datasets e100_H&OF, g196_H&OF and g196+e100_H&OF was obtained with network sizes of 2×1024, 2×1024 and 2×512, respectively. Fig. 10 show that the average validation errors of the H&OF models, too, were reduced by increasing the network size, but in general the errors were worse than those of the ST models even with large networks.

2) TEST RESULTS OF NON-HYBRID ANNs

The best models selected in the previous section were tested with the testing dataset. NRMSE and RMSE values were computed for each test case, and their average and maximum values are shown in Table 4 together with the simulation times and speed-up compared to FE simulation. First considering only the non-hybrid models, the training datasets e50 and w50, which have the smallest number of samples, showed similar average ($\approx 3.8\%$) and maximum ($\approx 30.9\%$) NRMSE values. Doubling the number of samples to a hundred made the corresponding average errors to halve to approximately 2%, but the maximum errors only dropped by one third. With the 200 case training datasets, the average NRMSEs improved slightly from the previous, but the maximum increased. The g196 training dataset had about the same number of cases as the e200 and the w200, but its average NRMSE (3.13%) was even worse than that of the hundred case datasets. When the number of training cases was further increased to 300, the corresponding average error decreased to 1.45% and 1.28%, respectively for the e300 and w300, with the latter being the lowest of the non-hybrid models. However, the maximum NRMSEs were high, respectively being 22.15% and 13.89%. The lowest maximum NRMSE value, 10.65%, was obtained with the g196+e100 model that also had the second best average NRMSE of 1.38% of the non-hybrid models.

The average (unnormalized) RMSEs of the validation cases shown in Table 4 are low in general, ranging between $0.009-0.061$ Nm. This was expected because the high errors are located in the operation area where the output torque magnitude is low. The relatively high maximum RMSE of the g196 model is due to the high errors also in the high torque magnitudes as Fig. 11 shows.

Comparing the model accuracies trained with an equal and non-equal number of samples in the different input amplitude ranges, the average and maximum NRMSEs of the w100…300 datasets were lower than the corresponding errors for the e datasets. This suggests that the model accuracy can be improved by placing more samples in the areas that seems to be harder model, without increasing the total number of samples. From this result, we can estimate that adaptive sampling could help.

3) TEST RESULTS OF HYBRID ANNs

The inputs of the models presented in Section IV-B2 included seven features extracted from the original three-phase input current signals as described in Section III-D and Fig. 7. Due to the large errors in the low current region, the H and H&OF versions of the e100, g196 and g196+e100 datasets were created. The models trained with these datasets are the previously described hybrid models with domain knowledge of torque behavior, with more details in Section III-D.

Results of the comparison between these three versions are shown in Table 4 and Fig. 11. Fig. 11 shows the test case NRMSEs case-by-case for the e100, g196 and g196 and their H and H&OF versions, plotted against the case input current amplitudes. The average and maximum NRMSE of the smallest dataset e100 were slightly improved with the H version as shown in Table 4. The improvement was even greater for the g196 dataset, as the average NRMSE more than halved to 1.39% and the maximum

TABLE 4. The average and maximum NRMSE and RMSE values and the simulation performance of the models trained on different datasets. The results were computed on the testing dataset. The reference FE simulation time was 146.5 s/case. The ANN computing times were measured as CPU time.

Dataset	NRMSE avg [%]	NRMSE max [%]	RMSE avg [Nm]	RMSE max [Nm]	ANN computing time avg [ms/case]	Times faster than FEM
e50	3.79	30.87	0.031	0.156	67.6	2,166
w50	3.81	30.87	0.045	0.322	68.0	2,154
e100	2.11	21.13	0.016	0.065	67.2	2,181
e100_H[a]	1.80	15.82	0.024	0.111	50.3	2,911
e100_H&OF[b]	4.74	27.12	0.045	0.157	66.8	2,194
w100	1.96	18.01	0.015	0.047	67.8	2,162
e200	1.92	26.78	0.012	0.036	56.9	2,574
w200	1.66	18.66	0.010	0.030	66.5	2,202
e300	1.45	22.15	0.009	0.030	55.7	2,631
w300	1.28	13.89	0.010	0.023	50.4	2,905
g196	3.13	13.76	0.061	0.258	50.5	2,901
g196_H[a]	1.39	4.82	0.030	0.112	56.9	2,575
g196_H&OF[b]	2.35	15.41	0.037	0.088	66.9	2,189
g196+e100	1.38	10.65	0.010	0.025	57.1	2,266
g196+e100_H[a]	1.14	5.23	0.017	0.050	67.8	2,161
g196+e100_H&OF[b]	1.76	4.28	0.030	0.071	56.1	2,613

[a] Hybrid model.
[b] Hybrid model using only the 3-phase current values in the input.

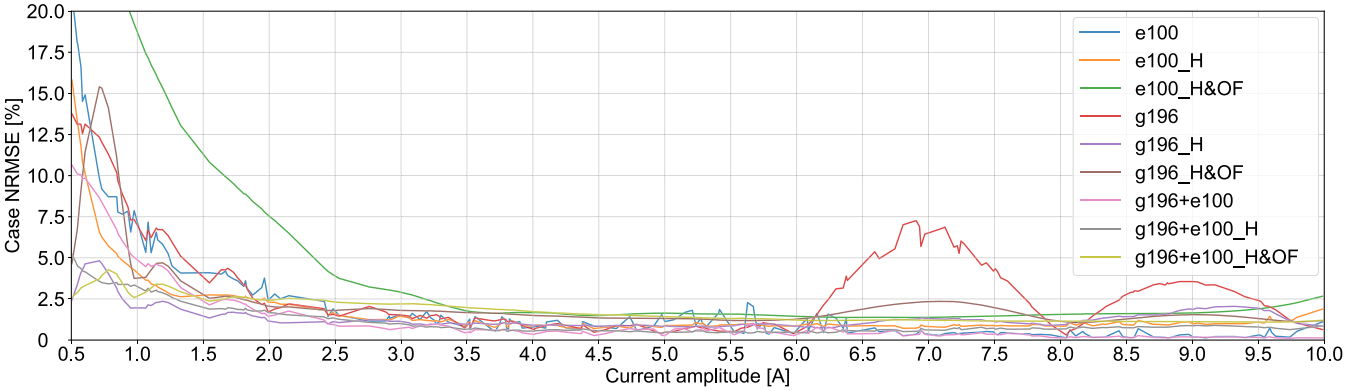

FIGURE 11. Test case NRMSEs of models trained on e100, g196 and g196+e100 datasets and their H and H&OF versions.

NRMSE decreased to one third, i.e. to 4.82%. The output normalization enhanced the g196+e100 results as well. Even though the average NRMSE of g196+e100_H decreased only slightly, the maximum NRMSE approximately halved from 10.65% to 5.23%. The results of H&OF show a worse average NRMSE than the hybrid and non-hybrid results for each of the three datasets. However, the maximum NRMSE of the g196+e100_H&OF model was lower than that of the H version (4.28% vs. 5.23%).

The case NRMSE values are generally worse with the low input current amplitudes compared to the high ones,[3] as shown in Fig. 11. The case NRMSEs of e100, g196 and g196+e100 models started to increase rapidly when the current amplitude was lower than 2.5 A. In addition, the g196 model accuracy decreased in between current amplitudes 6, 8 and 10 A, showing the weakness of grid sampling. In between those points, there are no samples in the g196 dataset. This could be avoided by increasing the grid density but it would quickly result in a much higher number of samples, especially if there were more input dimensions. The g196_H and g196+e100_H models' NRMSEs increased less at the low current amplitudes. Fig. 11 shows that the original input features (g196+e100_H&OF vs. g196+e100_H) are enough to achieve almost as good results on average as with the generated input features, but only with the hybrid model configuration.[4] Furthermore, executing more extensive hyperparameter optimization could enhance the accuracy. These results hint that in the attempt to increase the model prediction accuracy, not only should input feature generation be considered, but also, if applicable, the utilization of a simple hybrid model structure to manipulate the output variables.

Fig. 12 shows torque estimations of three ANN models trained with e100_H, g196_H and g196+e100_H, respectively. The upper plot shows one higher amplitude case in

[3] Using the mean squared logarithmic error (MSLE) loss function instead of MSE did not improve the results even though it accounts for the relative difference of the true and predicted values rather than the absolute difference.

[4] The time step lengths of the cases in the datasets were unified in this study as described in III-D. However, when using only the current values from the present time step in the model input, i.e. with the H&OF datasets, it is not necessary to perform the upsampling since no time-dependent input is considered. Leaving the upsampling step out from the workflow would reduce the number of samples and result in faster development of the ANN surrogate models.

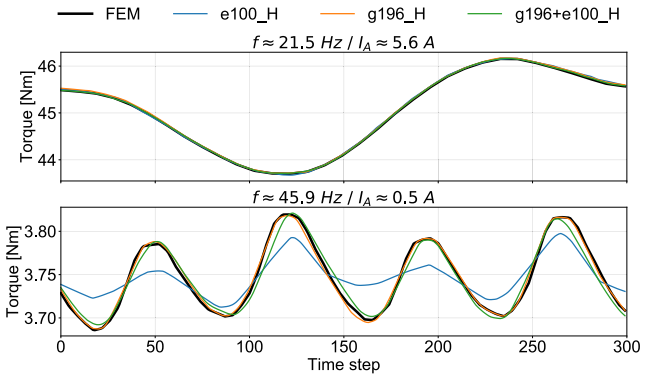

FIGURE 12. Torque estimations of ANN models trained with different datasets compared to the FE simulation results. The upper plot shows one example of a higher input current amplitude case. The bottom plot shows one of the lowest current amplitude cases.

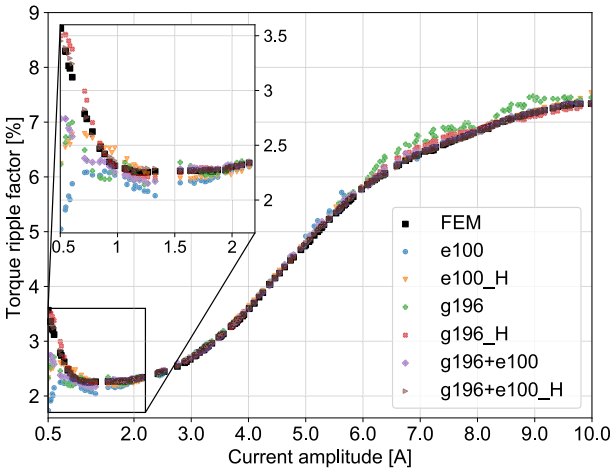

FIGURE 13. Torque ripple factor values computed from torque predictions made on the test dataset, including models trained on the e100, g196 and g196+e100 datasets and their hybrid (H) versions.

which each model performs well. The bottom plot, on the other hand, shows the weakness of randomized sampling on the outer edges of the input space, as the e100_H model has significantly worse accuracy than the other two models which were built using a dataset that contained samples on the outer edges. This can be also seen in Fig. 11, as the case NRMSEs of the three models are lower than 2.5% for current amplitudes higher than 2 A, but for lower amplitudes, the accuracy of e100_H begins to increase much more compared to the other two. The worst case NRMSE of g196_H and g196+e100_H are 4.82 and 5.23%, respectively. This together with the low average NRMSEs of these models (see Table 4) suggest that the accuracy is rather close to the FEM.

4) TORQUE RIPPLE FACTOR COMPARISON

In addition to the evaluation of the ANN model accuracies, their torque ripple estimation accuracies were compared. The numerical value of the torque ripple, the torque ripple factor [%], is defined as

$$t_r = \frac{T_{max} - T_{min}}{T_{avg}} \times 100, \tag{13}$$

where T_{max}, T_{min}, and T_{avg} are the maximum, minimum, and average values of the air gap torque time series, respectively. Torque ripple factor values were computed from (13) for each case in the test dataset from the predicted torque time series. The reference torque ripple factor curve was computed from the FE results. Due to the definition (13), the torque ripple factor has an high peak at $I = 0$ A, and rises rapidly from 1 A to 0.5 A. Above 1 A the ripple factor rises slowly due to other torque ripple terms. The ripple factor values of the ANN models are compared to the FE reference in Fig. 13. The torque ripple factors with the e100 model differ from the reference the most at the lowest input current amplitudes, where the computed values are too small, and the difference grow rapidly when moving towards 0.5 A. In fact, the g196 and g196+e100 models show similar behavior, however, the low amplitude offset is not as large. The accuracies of the hybrid models were better than the non-hybrid models which also shows in the torque ripple factor values as they are in general better. The torque ripple factor values are globally the closest to the reference with the g196_H and g196+e100_H models, which are also accurate in the lowest current amplitudes. The latter model of these two outperforms the former. The g196_H torque ripple factor modeling accuracy is poor between current amplitudes 6 to 8 and 8 to 10. For the same reason, the actual predictions are slightly worse – in that area, there are training cases only with current amplitudes of 6, 8 and 10 A, whereas in the g196+e100_H dataset there are a total of 40 cases.

C. SURROGATE DEVELOPMENT TIME AND COMPUTATIONAL EFFICIENCY

Improving the simulation efficiency compared to the FE model by utilizing surrogate models was the main motivation for this work. Thus, the simulation times of the FE model and the ANN models were compared. The reference simulation time of the FE model was 146.5 s/case, which is the average of the simulation time of 196 cases. The computing time of the ANN surrogate models varied between 50.3 to 68 ms/case, which makes the surrogates 2,911 to 2,154 times faster than the FE reference simulation, respectively (Table 4). The reported simulation times are measured in CPU time. The FE and ANN simulations were performed with different processors (Intel Xeon E5-2640 v3 and E5-2690 v4, respectively), with one processor generation difference but the same processor base frequency, while the FE simulations were done using the slower processor. However, the computational efficiency comparison is quite fair, as the real processor performance difference is small.

Table 5 shows the overall times to develop a surrogate model with the training datasets e100_H, g196_H and g196+e100_H. The FE simulation times for the training datasets were 4.1 h, 8 h and 12.1 h, respectively. The corresponding simulation time of the validation and testing datasets were 6.1 h and 7.7 h, respectively. The data generation times and the surrogate model development times

TABLE 5. Total times to develop a surrogate model including data generation with FE simulation and training the ANN models. The FE simulation times include simulations of training, validation, and testing datasets.

Dataset	FE simulation [h]	ANN training [h]	Total [h]
e100_H	17.9	2.2	20.1
g196_H	21.8	5.5	27.3
g196+e100_H	25.9	5.7	31.6

vary between the datasets, as the number of cases are different. It should be noticed that the computational times were calculated by assuming a sequential execution without any parallelization applied. As the simulation of different cases in FEM are independent of each other, and as long as a random or a grid search is used in the ANN hyperparameter optimization, it is technically possible to use a hundred similar computational units, for example, in both steps and speed up the surrogate model development process by almost a 100-fold.

The hyperparameter optimization with the e100_H, g196_H and g196+e100_H datasets took 2.2 h, 5.5 h and 5.7 h, respectively. These times correspond to an additional 53, 135, or 140 case simulations in FEM. It should be noted that PMSM design optimization could require many more FE simulations to explore the design space than were done in this case study to the develop the surrogates. Hence, after developing the surrogate, the simulation evaluations would be significantly cheaper computationally.

Even more important than the above-described model development times are the model run-time computation times. Let's compare the two most accurate models based on comparisons of the different datasets in section IV-B2, namely the g196+e100_H and g196+e100_H&OF models. The g196+e100_H computation time was 67.8 ms/case whereas g196+e100_H&OF took 56.1 ms/case, meaning a 20% faster simulation performance for the latter. The increased performance of the H&OF version is due to the lower number of parameters in the ANN structure compared to the H version. Altogether, even though the average NRMSE of the g196+e100_H&OF model is slightly worse than that of the H version, the lower maximum NRMSE and higher simulation performance favors g196+e100_H&OF for selection as the best ANN surrogate model developed here.

V. DISCUSSION AND CONCLUSION

In this article, we have reviewed the surrogate modeling concept and its existing applications in the EM domain, and demonstrated how to utilize machine learning in surrogate modeling. We have presented a workflow to create a surrogate model of a physics-based simulation model of an electrical machine, compared two selected ML-based models, namely ANN and GBDT, compared different data sampling approaches for producing the data needed in the creation of the surrogate models, and compared the performance of the surrogate models with the physics-based FE simulation of an EM. The EM type selected for this study was an IPMSM. The physics-based simulations were done with a 2D FE model of the EM, varying the machine input current frequency and amplitude as the simulation parameters. The output of the simulations was the air gap torque of the EM. The motivation for the work was to study whether the surrogate models could be used to replace the physics-based simulation models in certain applications, to determine their performance and the effort needed to generate the surrogates.

The comparison between the GBDT and ANN models showed that even though developing a GBDT surrogate was faster than an ANN surrogate, its inference performance and accuracy were not as good as the ANN approach. In addition, the smooth output behavior of ANN favored it over GBDT in EM applications. Due to this, we continued to the data sampling experiments with the ANN models. These experiments showed that both grid and randomized sampling methods can provide good results when modeling the torque behavior of a PMSM, especially when a simple hybrid model structure is utilized. The best accuracy and torque ripple factor estimation on the test dataset were obtained by training the ANN model with a training dataset which combined grid and randomized sampling. The average NRMSE of the best hybrid model was 1.8% in the test cases. Compared to the torque estimation of a projection-based surrogate for an identical electrical machine design in [22] as used here, the ANN surrogate seems to have smoother torque curves, and more accurate estimations, at least for higher currents. The combined training dataset most likely provided better results due to having samples at the outer edges of the input space, including low currents.[5] The randomized sampling led to better model accuracy with fewer samples than grid sampling, apart from the low currents. Using a denser grid is not reasonable, as the number of samples rises quickly. A non-adaptive design of experiments was sufficient to develop an accurate surrogate model in this study, but in cases with more input dimensions, using an adaptive sampling method could potentially be a better choice.

In Section I, the nature of computational methods was discussed. It was pointed out that different methods tend to be either fast but inaccurate or accurate but computationally expensive. The introduced method of using ANNs seems to tackle both the accuracy and efficiency. The drawback of the method is in the ANN development, as the computational cost occurs in the data production and the training phases. On the other hand, the surrogate model can be trained offline and the key benefit of developing one is that a fast and sufficiently accurate surrogate potentially enables new applications for the simulation model, for which the FE simulation would be too slow. Such applications could be include machine control, condition monitoring and fault diagnosis, for example. Nevertheless, the ANN surrogates show potential in accelerating

[5]The need for many samples in the input space borders may be due to the unfortunately selected low range of current, namely 0.5 A. The choice limits the number of samples in the low current region where cogging torque dominates.

simulation. The performance of the ANN surrogate models trained with different datasets were 2,154 to 2,911 times faster than the physics-based FE PMSM simulation, the computing times of the ANN surrogates being in the scale of tens of milliseconds for a case compared to about 150 seconds for the FE simulation. The drawback of the time needed for surrogate development becomes smaller when the approach is applied, for example, in industrial series production of machines, as the fast surrogate can be used in numerous produced machine units by employing transfer learning. The introduced application of an EM is a good example of a general approach to using simulated data in machine learning of surrogate models, since the computational performance of a physics-based FE simulation is far from real-time, even when efficient computers are used. It should be noticed that the model used for the FE simulation was relatively coarse, being reduced to 2D, with a modest number of nodes and elements, and no eddy current or losses.

The proposed ANN surrogate model is well-suited for a system-level model or digital twin applications, as it represents the system with a good accuracy. LUTs are one alternative for ANN surrogates, but the accuracy and computational costs of LUTs become the main constraints when the number of input parameters increases, whereas the ANN surrogate performance does not deteriorate as much. Regarding the computational time, one evaluation for optimal g196+e100_H&OF sampling was about 42 μs with Python, which is fast enough for a controller with a 20 kHz switching frequency. Therefore, the proposed model can be used in real-time EM control, for example as a torque observer based on the measured currents. Similar ANN surrogate can be used as a flux observer by modifying the model to have the flux linkages components as outputs. In addition, the proposed ANN surrogate can be used in electrical machine design, as an alternative to circuit-based analytical models to estimate the initial designs and then use the FE analysis only for the final fine-tuning of the design. The ANN surrogate model type developed here is suitable for use as a global or local surrogate in the surrogate model-based optimization of EM design. Potentially, further improvements to the ANN accuracy could be obtained by performing more extensive hyperparameter optimizations or by increasing the number of FE simulation data points generated for training.

One advantage, especially from the industrial applications point of view, of using ANNs and machine learning for creating surrogate models is the ability to generalize the approach and semi-automate their creation process. Even though domain knowledge is needed to specify the input variable domain, i.e. the value ranges that the input variables can have, running the FE simulations can be parameterized and the simulations parallelized. The produced data can be directly used in ANN training and the accuracy requirement of the surrogate can be set parametrically. Thus, the development of a surrogate model that fulfills the requirements can be mainly automated. The generality of ML as a method, and the flexibility of ANNs when it comes to the number of inputs and outputs, the nature of the ANN response, and the accuracy make it an interesting tool for surrogate modeling. On the other hand, the black-box nature of ANNs hinders the explainability of the predictions. Nevertheless, the clear benefits of the approach together with the fast progress in enabling technologies speak for investing more in research and development.

ACKNOWLEDGMENT

The authors would like to thank Dr. Victor Mukherjee (ABB, Finland), Dr. Jenni Pippuri-Mäkeläinen, Dr. Jussi Kiljander, and Milla Vehviläinen (VTT Technical Research Centre of Finland) for their valuable comments that greatly improved the manuscript.

REFERENCES

[1] K. McBride and K. Sundmacher, "Overview of surrogate modeling in chemical process engineering," *Chem. Ingenieur Technik*, vol. 91, no. 3, pp. 228–239, Mar. 2019.
[2] L. Ljung, *System Identification—Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.
[3] R. Isermann and M. Münchhof, *Identification of Dynamic Systems: An Introduction With Applications*, 1st ed. Berlin, Germany: Springer-Verlag, 2011.
[4] C. de Villemagne and R. E. Skelton, "Model reductions using a projection formulation," *Int. J. Control*, vol. 46, no. 6, pp. 2141–2169, 1987.
[5] T. D. Robinson, M. S. Eldred, K. E. Willcox, and R. Haimes, "Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping," *AIAA J.*, vol. 46, no. 11, pp. 2814–2822, Nov. 2008.
[6] A. Natekin and A. Knoll, "Gradient Boosting Machines, a Tutorial," *Front. Neurorobot.*, vol. 7, Dec. 2013.
[7] K. Hameyer and R. Belmans, *Numerical Modelling and Design of Electrical Machines and Devices*. Southampton, U.K.: WIT Press, 1999.
[8] J. Keranen, P. Ponomarev, J. Pippuri, P. Raback, M. Lyly, and J. Westerlund, "Parallel performance of multi-slice finite-element modeling of skewed electrical machines," *IEEE Trans. Magn.*, vol. 53, no. 6, pp. 1–4, Jun. 2017.
[9] X. Wang, D. Liu, D. Lahaye, H. Polinder, and J. A. Ferreira, "Finite element analysis and experimental validation of eddy current losses in permanent magnet machines with fractional-slot concentrated windings," in *Proc. 19th Int. Conf. Electr. Mach. Syst. (ICEMS)*, Nov. 2016, pp. 1–6.
[10] J. L. Lumley, "The structure of inhomogeneous turbulent flows," in *Atmospheric Turbulence and Radio Wave Propagation*, A. Yaglom and V. Tatarski, Eds. Moscow, Russia: Nauka, 1967, pp. 166–178.
[11] W. H. A. Schilders, "The need for novel model order reduction techniques in the electronics industry," in *Model Reduction for Circuit Simulation*, vol. 74, P. Benner, M. Hinze, and E. J. W. ter Maten, Eds. Dordrecht, The Netherlands: Springer, 2011, pp. 3–23.
[12] G. Kerschen, J.-C. Golinval, A. F. Vakakis, and L. A. Bergman, "The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview," *Nonlinear Dyn.*, vol. 41, nos. 1–3, pp. 147–169, Aug. 2005.
[13] T. J. Snowden, P. H. van der Graaf, and M. J. Tindall, "Methods of model reduction for large-scale biological systems: A survey of current methods and trends," *Bull. Math. Biol.*, vol. 79, no. 7, pp. 1449–1486, Jul. 2017.
[14] A. C. Antoulas, D. Sorensen, and S. Gugercin, "A survey of model reduction methods for large-scale systems," *Contemp. Math.*, vol. 280, pp. 193–219, Oct. 2001.
[15] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, 1st ed. Cham, Switzerland: Springer, 2015.
[16] S. Volkwein, "Optimal control of a phase-field model using proper orthogonal decomposition," *ZAMM, J. Appl. Math. Mech.*, vol. 81, no. 2, pp. 83–97, 2001.
[17] A. Nouy, "*A priori* model reduction through proper generalized decomposition for solving time-dependent partial differential equations," *Comput. Methods Appl. Mech. Eng.*, vol. 199, nos. 23–24, pp. 1603–1626, Apr. 2010.
[18] D. Ryckelynck, "*A priori* hyperreduction method: An adaptive approach," *J. Comput. Phys.*, vol. 202, no. 1, pp. 346–366, 2005.

[19] W. Z. Lin, Y. J. Zhang, and E. P. Li, "Proper orthogonal decomposition in reduced order model generation for microwave problem," in *Proc. 17th Int. Zurich Symp. Electromagn. Compat.*, Singapore, 2006, pp. 223–226.
[20] S. Clenet, T. Henneron, and N. Ida, "Reduction of a finite-element parametric model using adaptive POD methods—Application to uncertainty quantification," *IEEE Trans. Magn.*, vol. 52, no. 3, pp. 1–4, Mar. 2016.
[21] A. Pierquin, T. Henneron, S. Clenet, and S. Brisset, "Model-order reduction of magnetoquasi-static problems based on POD and Arnoldi-based Krylov methods," *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1–4, Mar. 2015.
[22] M. Farzamfar, A. Belahcen, P. Rasilo, S. Clenet, and A. Pierquin, "Model order reduction of electrical machines with multiple inputs," *IEEE Trans. Ind. Appl.*, vol. 53, no. 4, pp. 3355–3360, Jul. 2017.
[23] T. Sato, Y. Sato, and H. Igarashi, "Model order reduction for moving objects: Fast simulation of vibration energy harvesters," *COMPEL-Int. J. Comput. Math. Electr. Electron. Eng.*, vol. 34, no. 5, pp. 1623–1636, Sep. 2015.
[24] M. N. Albunni, V. Rischmuller, T. Fritzsche, and B. Lohmann, "Model-order reduction of moving nonlinear electromagnetic devices," *IEEE Trans. Magn.*, vol. 44, no. 7, pp. 1822–1829, Jul. 2008.
[25] T. Henneron and S. Clenet, "Model order reduction of non-linear magnetostatic problems based on POD and DEI methods," *IEEE Trans. Magn.*, vol. 50, no. 2, pp. 33–36, Feb. 2014.
[26] M. F. Far, F. Martin, A. Belahcen, P. Rasilo, and H. A. A. Awan, "Real-time control of an IPMSM using model order reduction," *IEEE Trans. Ind. Electron.*, vol. 68, no. 3, pp. 2005–2014, Mar. 2021.
[27] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.
[28] M. Cavazzuti, *Optimization Methods: From Theory to Design*. Berlin, Germany: Springer-Verlag, 2013.
[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[30] A. Khadilkar, J. Wang, and R. Rai, "Deep learning–based stress prediction for bottom-up SLA 3D printing process," *Int. J. Adv. Manuf. Technol.*, vol. 102, nos. 5–8, pp. 2555–2569, Jun. 2019.
[31] B. N. Fetene, R. Shufen, and U. S. Dixit, "FEM-based neural network modeling of laser-assisted bending," *Neural Comput. Appl.*, vol. 29, no. 6, pp. 69–82, Mar. 2018.
[32] R. Kishore, R. Mahajan, and S. Priya, "Combinatory finite element and artificial neural network model for predicting performance of thermoelectric generator," *Energies*, vol. 11, no. 9, p. 2216, Aug. 2018.
[33] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3147–3155.
[34] C. Gu, "Model order reduction of nonlinear dynamical systems," Ph.D. dissertation, EECS Dept., Univ. California, Berkeley, CA, USA, 2012.
[35] M. Lovera, M. Bergamasco, and F. Casella, "LPV modelling and identification: An overview," in *Robust Control and Linear Parameter Varying Approaches*, O. Sename, P. Gaspar, and J. Bokor, Eds. Berlin, Germany: Springer, 2013, pp. 3–24.
[36] S. Seher-Weiss, "Identification of nonlinear aerodynamic derivatives using classical and extended local model networks," *Aerosp. Sci. Technol.*, vol. 15, no. 1, pp. 33–44, Jan. 2011.
[37] M. von Stosch, R. Oliveira, J. Peres, and S. F. D. Azevedo, "Hybrid semi-parametric modeling in process systems engineering: Past, present and future," *Comput. Chem. Eng.*, vol. 60, pp. 86–101, Jan. 2014.
[38] S. Haykin, *Neural Networks and Learning Machines*, vol. 3, 3rd ed. London, U.K.: Pearson Prentice-Hall, 2008.
[39] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[40] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.
[41] L. Prechelt, *Early Stopping—But When?* G. B. Orr and K.-R. Muller, Eds. Berlin, Germany: Springer, 1998.
[42] R. May, G. Dandy, and H. Maier, "Review of input variable selection methods for artificial neural networks," in *Artificial Neural Networks—Methodological Advances and Biomedical Applications*, K. Suzuki, Ed. Rijeka, Croatia: InTech, 2011.
[43] V. Perez, J. Renaud, and L. Watson, "Adaptive experimental design for construction of response surface approximations," in *Proc. 19th AIAA Appl. Aerodynamics Conf.*, Jun. 2001, pp. 1–12.
[44] K. K. Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti, "Surrogate-based methods for black-box optimization," *Int. Trans. Oper. Res.*, vol. 24, no. 3, pp. 393–424, May 2017.
[45] Y. Ji, S. Kim, and W. Xu, "A new framework for combining global and local methods in black box optimization," *Optim. Online*, vol. 3977, pp. 1–32, Jul. 2013. [Online]. Available: http://www.optimization-online.org/DB_FILE/2013/07/3977.pdf
[46] Z. Tan, X. Song, W. Cao, Z. Liu, and Y. Tong, "DFIG machine design for maximizing power output based on surrogate optimization algorithm," *IEEE Trans. Energy Convers.*, vol. 30, no. 3, pp. 1154–1162, Sep. 2015.
[47] D.-K. Lim, K.-P. Yi, S.-Y. Jung, H.-K. Jung, and J.-S. Ro, "Optimal design of an interior permanent magnet synchronous motor by using a new surrogate-assisted multi-objective optimization," *IEEE Trans. Magn.*, vol. 51, no. 11, pp. 1–4, Nov. 2015.
[48] S. Giurgea, H. S. Zire, and A. Miraoui, "Two-stage surrogate model for finite-element-based optimization of permanent-magnet synchronous motor," *IEEE Trans. Magn.*, vol. 43, no. 9, pp. 3607–3613, Sep. 2007.
[49] F. Gillon and P. Brochet, "Screening and response surface method applied to the numerical optimization of electromagnetic devices," *IEEE Trans. Magn.*, vol. 36, no. 4, pp. 1158–1162, Jul. 2000.
[50] L. Jolly, M. A. Jabbar, and L. Qinghua, "Design optimization of permanent magnet motors using response surface methodology and genetic algorithms," in *INTERMAG ASIA, Dig. IEEE Int. Magn. Conf.*, Apr. 2005, vol. 41, no. 10, p. 46.
[51] L. Jolly, M. A. Jabbar, and L. Qinghua, "Optimization of the constant power speed range of a saturated permanent magnet synchronous motor," in *Proc. IEEE Int. Conf. Electr. Mach. Drives*, May 2005, vol. 42, no. 4, pp. 1024–1030.
[52] H. M. Hasanien, A. S. Abd-Rabou, and S. M. Sakr, "Design optimization of transverse flux linear motor for weight reduction and performance improvement using response surface methodology and genetic algorithms," *IEEE Trans. Energy Convers.*, vol. 25, no. 3, pp. 598–605, Sep. 2010.
[53] S. Vivier, F. Gillon, and P. Brochet, "Optimization techniques derived from experimental design method and their application to the design of a brushless direct current motor," *IEEE Trans. Magn.*, vol. 37, no. 5, pp. 3622–3626, Sep. 2001.
[54] X. Gao, T.-S. Low, S. Chen, and Z. Liu, "Structural robust design for torque optimization of BLDC spindle motor using response surface methodology," *IEEE Trans. Magn.*, vol. 37, no. 4, pp. 2814–2817, Jul. 2001.
[55] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. Van de Walle, and S. Van Hoecke, "Convolutional neural network based fault detection for rotating machinery," *J. Sound Vibrat.*, vol. 377, pp. 331–345, Sep. 2016.
[56] I. Martin-Diaz, D. Morinigo-Sotelo, O. Duque-Perez, and R. J. Romero-Troncoso, "An experimental comparative evaluation of machine learning techniques for motor fault diagnosis under various operating conditions," *IEEE Trans. Ind. Appl.*, vol. 54, no. 3, pp. 2215–2224, may/Jun. 2018.
[57] L. Wen, X. Li, L. Gao, and Y. Zhang, "A new convolutional neural network-based data-driven fault diagnosis method," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5990–5998, Jul. 2018.
[58] J. S. L. Senanayaka, H. Van Khang, and K. G. Robbersmyr, "Online fault diagnosis system for electric powertrains using advanced signal processing and machine learning," in *Proc. 23rd Int. Conf. Electr. Mach. (ICEM)*, Sep. 2018, pp. 1932–1938.
[59] J. C. Quiroz, N. Mariun, M. R. Mehrjou, M. Izadi, N. Misron, and M. A. M. Radzi, "Fault detection of broken rotor bar in LS-PMSM using random forests," *Measurement*, vol. 116, pp. 273–280, Feb. 2018.
[60] R. Liu, G. Meng, B. Yang, C. Sun, and X. Chen, "Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1310–1320, Jun. 2017.
[61] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mech. Syst. Signal Process.*, vols. 72–73, pp. 303–315, May 2016.
[62] H. A. A. Awan, Z. Song, S. E. Saarakkala, and M. Hinkkanen, "Optimal torque control of saturated synchronous motors: Plug-and-play method," *IEEE Trans. Ind. Appl.*, vol. 54, no. 6, pp. 6110–6120, Nov. 2018.
[63] M. Liu, K. W. Chan, J. Hu, W. Xu, and J. Rodriguez, "Model predictive direct speed control with torque oscillation reduction for PMSM drives," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 4944–4956, Sep. 2019.
[64] H. Ren, W. Song, and Z. Ruan, "Model predictive control of PMSM considering online parameter identification," in *Proc. IEEE 3rd Int. Electr. Energy Conf. (CIEEC)*, Sep. 2019, pp. 1307–1311.

[65] Z. Xu, Z. Wang, X. Wang, and M. Cheng, "Predictive current control method for dual three-phase PMSM drives with reduced switching frequency and low-computation burden," *IET Electr. Power Appl.*, vol. 14, no. 4, pp. 668–677, Apr. 2020.
[66] S. Li, H. Won, X. Fu, M. Fairbank, D. C. Wunsch, and E. Alonso, "Neural-network vector controller for permanent-magnet synchronous motor drives: Simulated and hardware-validated results," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3218–3230, Jul. 2020.
[67] J. Yu, P. Shi, S. Member, W. Dong, B. Chen, and C. Lin, "Brief papers permanent magnet synchronous motors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 640–645, 2015.
[68] Y.-B. Yan, J.-N. Liang, T.-F. Sun, J.-P. Geng, Gang-Xie, and D.-J. Pan, "Torque estimation and control of PMSM based on deep learning," in *Proc. 22nd Int. Conf. Electr. Mach. Syst. (ICEMS)*, Aug. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8921886
[69] D. E. Pinto, A.-C. Pop, J. Kempkes, and J. Gyselinck, "Dq0-modeling of interior permanent-magnet synchronous machines for high-fidelity model order reduction," in *Proc. Int. Conf. Optim. Electr. Electron. Equip. (OPTIM) Intl Aegean Conf. Electr. Mach. Power Electron. (ACEMP)*, May 2017, pp. 357–363.
[70] K. Drobnic, L. Gasparin, and R. Fiser, "Fast and accurate model of interior permanent-magnet machine for dynamic characterization," *Energies*, vol. 12, no. 5, p. 783, Feb. 2019.
[71] M. A. Rahman, "History of interior permanent magnet motors," *IEEE Ind. Appl. Mag.*, vol. 19, no. 1, pp. 10–15, Jan./Feb. 2013.
[72] V. Mukherjee, T. Martinovski, A. Szucs, J. Westerlund, and A. Belahcen, "Improved analytical model of induction machine for digital twin application," in *Proc. Int. Conf. Elect. Mach. (ICEM)*, 2020, pp. 183–189.
[73] V. Ostović, *Dynamics of Saturated Electric Machines*. New York, NY, USA: Springer-Verlag, 1989.
[74] J. Perho, "Reluctance network for analysing induction machines," Ph.D. dissertation, Dept. Elect. Commun. Eng., Helsinki Univ. Technol., Espoo, Finland, 2002.
[75] A. Arkkio, "Analysis of induction motors based on the numerical solution of the magnetic field and circuit equations," Ph.D. dissertation, Dept. Elect. Commun. Eng., Helsinki Univ. Technol., Espoo, Finland, 1987.
[76] H. W. de Kock, A. J. Rix, and M. J. Kamper, "Optimal torque control of synchronous machines based on finite-element analysis," *IEEE Trans. Ind. Electron.*, vol. 57, no. 1, pp. 413–419, Jan. 2010.
[77] F. Soares and P. J. Costa Branco, "Simulation of a 6/4 switched reluctance motor based on MATLAB/simulink environment," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 3, pp. 989–1009, Jul. 2001.
[78] L. Queval and H. Ohsaki, "Nonlinear abc-model for electrical machines using N-D lookup tables," *IEEE Trans. Energy Convers.*, vol. 30, no. 1, pp. 316–322, Mar. 2015.
[79] *Elmer, An Open Source Finite Element Method Software for Multiphysical Problems*. Accessed: Nov. 17, 2020. [Online]. Available: https://www.csc.fi/web/elmer
[80] J. Ruokolainen, M. Malinen, P. Råback, T. Zwinger, A. Pursula, and M. Byckling. (2016). *ElmerSolver Manual*. Accessed: Nov. 17, 2020. [Online]. Available: http://www.funet.fi/pub/sci/physics/elmer/doc/ElmerSolverManual.pdf
[81] M. Farzam Far, F. Martin, A. Belahcen, L. Montier, and T. Henneron, "Orthogonal interpolation method for order reduction of a synchronous machine model," *IEEE Trans. Magn.*, vol. 54, no. 2, pp. 1–6, Feb. 2018.
[82] *FreeCAD, a 3D CAD/CAE Parametric Modeling Application*. Accessed: Nov. 17, 2020. [Online]. Available: https://www.freecadweb.org/
[83] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *Int. J. Numer. Methods Eng.*, vol. 79, no. 11, pp. 1309–1331, Sep. 2009.
[84] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Accessed: Nov. 17, 2020. [Online]. Available: http://tensorflow.org/
[85] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 9, 2010, pp. 249–256.
[86] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[87] S. L. Smith, P. J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–11.
[88] A. Devarakonda, M. Naumov, and M. Garland, "AdaBatch: Adaptive batch sizes for training deep neural networks," 2017, *arXiv:1712.02029*. [Online]. Available: http://arxiv.org/abs/1712.02029

MIKKO TAHKOLA received the B.Eng. degree in energy technology from the Oulu University of Applied Sciences in 2017, and M.Sc. degree (Tech.) in process engineering from the University of Oulu in 2019. He is currently a Research Scientist with the AI-Aided Systems Engineering Research Team, VTT Technical Research Centre of Finland. His research interests are related to developing ML-based tools for engineering and ML-based surrogate modeling.

JANNE KERÄNEN received the M.Sc. (Tech.) and D.Sc. (Tech.) degrees from the Tampere University of Technology, Finland, in 2001 and 2011, respectively. His background is in electrical engineering, especially in electromagnetic computational methods. He is currently a Senior Scientist with the Electrical Powertrains and Storage Research Team, VTT Technical Research Centre of Finland, where he is also working as a Principal Investigator, leading the research spearhead in computational machine dynamics. His special research interests include computational methods, electrical machines, different data- and ML-based surrogate models of electrical powertrains, and electromagnetic modeling and simulation methods.

DENIS SEDOV received the B.Sc. degree from Petrozavodsk State University in 2014 and the M.Sc. (Tech.) degree from the Lappeenranta University of Technology in 2017. He is currently pursuing the Ph.D. degree with Aalto University, Finland. His research focuses on large-scale clustering methods and their applications.

MEHRNAZ FARZAM FAR received the M.Sc. (Tech.) and D.Sc. (Tech.) degrees from Aalto University, Espoo, Finland, in 2014 and 2019, respectively. She joined the VTT Technical Research Centre of Finland in 2019, where she is currently a Research Scientist with the Electrical Powertrains and Storage Team. Her research interests include numerical modeling of electrical machines, model order reduction, and machine learning techniques.

JUHA KORTELAINEN received the M.Sc. (Tech.) degree from the Helsinki University of Technology in 1995 and the D.Sc. (Tech.) degree in virtual design on the subject of semantic data model for multibody system modeling from the Lappeenranta University of Technology in 2011. He was an Engineering Analyst with a private consulting company. From 1998 to 2001, he was a Researcher with the Helsinki University of Technology, focusing on computational research of internal combustion engines. Since 2002, he has been with the VTT Technical Research Centre of Finland as a Research Scientist, a Senior Scientist, the Research Team Leader, and a Principal Scientist. Since 2014, his research work has been focused on the process industry domain. He is also responsible for guiding the research in engineering design at VTT Technical Research Centre of Finland. He is currently a Principal Scientist and a Principal Investigator with the VTT Technical Research Centre of Finland, focusing on systems engineering and simulation research, and digital engineering design. His background is in mechanical engineering and computational product development.